Министерство образования Республики Беларусь

УО «Брестский государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №2**

По дисциплине: “Языки программирования”

Тема: “Наследование и виртуальные функции”**Вариант №1**

**Выполнил**:

студент 2 курса группы ПО-7 Белев С. В.

**Проверила:**

Дряпко А. В.

Брест 2021

**Цель:**

Получить практические навыки создания иерархии классов и использования статических компонентов класса.

**Постановка задачи:**

Написать программу, в которой создается иерархия классов. Включить полиморфные объекты в связанный список, используя статические компоненты класса. Показать использование виртуальных функций.

Порядок выполнения работы:

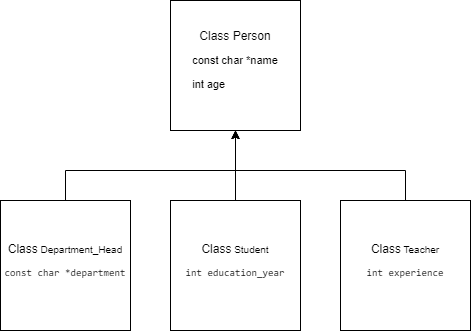
1. Определить иерархию классов (в соответствии с вариантом).
2. Определить в классе статическую компоненту - указатель на начало связанного списка объектов и статическую функцию для просмотра списка.
3. Реализовать классы.
4. Написать демонстрационную программу, в которой создаются
5. объекты различных классов и помещаются в список, после чего список просматривается.
6. Сделать соответствующие методы не виртуальными и посмотреть, что будет.
7. Реализовать вариант, когда объект добавляется в список при создании, т.е. в конструкторе (смотри пункт 6 следующего раздела).

**Вариант задания:**

Перечень классов:

1) студент, преподаватель, персона, завкафедрой

**Иерархия классов:**



**Код программы:**

**main.cpp**

#include "Header.h"

int person::count = 0;

person\*\* person::persons = nullptr;

int main() {

student michael("Michael", 19, 2);

michael.add();

teacher dunnig("Dunnig", 44, 12);

teacher alex("Alex", 29, 5);

department\_head oleg("Oleg", 53, "ALOE");

person::show();

}

**classes.h**

#include <string>

#include <iostream>

using namespace std;

class person {

public:

const char\* name;

int age;

static int count;

static person\*\* persons;

person(const char\* name, int age, bool isadd = true);

~person();

virtual void print() = 0;

virtual void add();

static void show();

};

class student : public person {

protected:

int education\_year;

public:

student(const char\* name, int age, int education\_year);

~student();

void print();

};

class teacher : public person {

protected:

int experience;

public:

teacher(const char\* name, int age, int experience);

~teacher();

void print();

};

class department\_head : public person {

protected:

const char\* department;

public:

department\_head(const char\* name, int age, const char\* department);

~department\_head();

void print();

};

**Class.cpp**

#include "Header.h"

person::person(const char\* name, int age, bool isadd) :

name(name), age(age) {

if (isadd) add();

cout << "Person " << name << " created\n";

}

person :: ~person() {

cout << "Person " << name << " deleted\n";

}

void person::add() {

person\*\* temp = persons;

persons = new person \* [count + 1];

for (int i = 0; i < count; i++)

persons[i] = temp[i];

persons[count] = this;

count++;

}

void person::show() {

for (int i = 0; i < count; i++)

persons[i]->print();

}

student::student(const char\* name, int age, int education\_year) :

person(name, age, false), education\_year(education\_year) {

cout << "Student " << name << " created\n";

}

student :: ~student() {

cout << "Student " << name << " deleted\n";

}

void student::print() {

cout << "Student name: " << name << " Age: " << age << " Education year: " << education\_year << "\n";

}

teacher::teacher(const char\* name, int age, int experience) :

person(name, age, true), experience(experience) {

cout << "Teacher " << name << " created\n";

}

teacher :: ~teacher() {

cout << "Student " << name << " deleted\n";

}

void teacher::print() {

cout << "Teacher name: " << name << " Age: " << age << " Experience: " << experience << "\n";

}

department\_head::department\_head(const char\* name, int age, const char \* department) :

person(name, age, true), department(department) {

cout << "Head of the Department " << name << " created\n";

}

department\_head :: ~department\_head() {

cout << "Head of the Department " << name << " deleted\n";

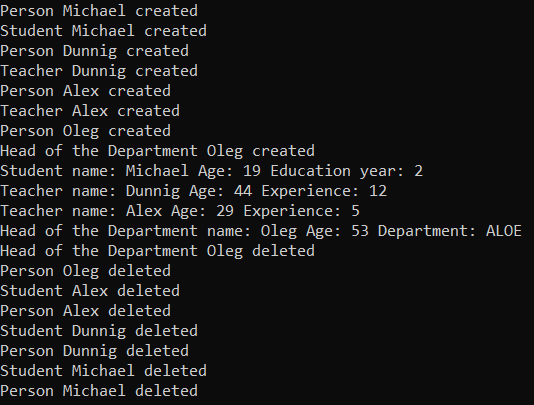
}

void department\_head::print() {

cout << "Head of the Department name: " << name << " Age: " << age << " Department: " << department << "\n";

}

**Результат программы:**



**Вывод:**

Я получил практические навыки создания иерархии классов и использования статических компонентов класса.